Задание для WPF-приложения на тему "yield return в C#"

Цель задания: Закрепить практические навыки работы с ключевым словом yield return в C# и научиться использовать его в WPF-приложениях для создания ленивых итераторов.

Условие задания:

Разработайте WPF-приложение, которое будет использовать метод с yield return для ленивой генерации последовательности чисел, а затем отображать их в **ListBox** в пользовательском интерфейсе. Пользователь должен иметь возможность указать диапазон чисел (начальное и конечное значение) и нажать кнопку, чтобы начать процесс генерации чисел. Числа должны отображаться по мере их генерации.

Требования к реализации:

1. **Класс для генерации чисел с использованием yield return**:
    - Создайте метод, который будет генерировать числа в диапазоне от начального до конечного с использованием ключевого слова yield return. Например, метод GenerateNumbers(int start, int end) генерирует числа от start до end.
    - Метод должен возвращать IEnumerable<int>, что позволит использовать yield return для пошаговой генерации значений.
2. **Интерфейс WPF**:
    - Создайте два поля ввода (TextBox) для указания начального и конечного значений диапазона.
    - Добавьте кнопку для запуска генерации чисел.
    - Используйте **ListBox** для отображения сгенерированных чисел.
    - *Добавьте ProgressBar, чтобы показывать процесс генерации чисел (например, в зависимости от оставшихся чисел).*
3. **Обработка событий**:
    - При нажатии кнопки «Генерировать» значения из полей должны быть прочитаны и переданы в метод генерации чисел.
    - По завершению генерации пользователю выводится сообщение о завершении.

- *Числа должны отображаться по одному в **ListBox**, и это должно происходить с задержкой, чтобы продемонстрировать ленивую генерацию. (Task.Delay)*

Критерии оценки:

1. **Работоспособность**: Приложение должно корректно генерировать числа и отображать их в ListBox с задержкой.
2. **Использование yield return**: Метод генерации чисел должен использовать yield return.
3. **Удобство интерфейса**: Пользовательский интерфейс должен быть понятным, функциональным и интерактивным.
4. ***Асинхронность**: Генерация чисел должна происходить с задержкой, чтобы продемонстрировать ленивую генерацию.*